

Shift-up: Where is the puck in GenAI-native software engineering?

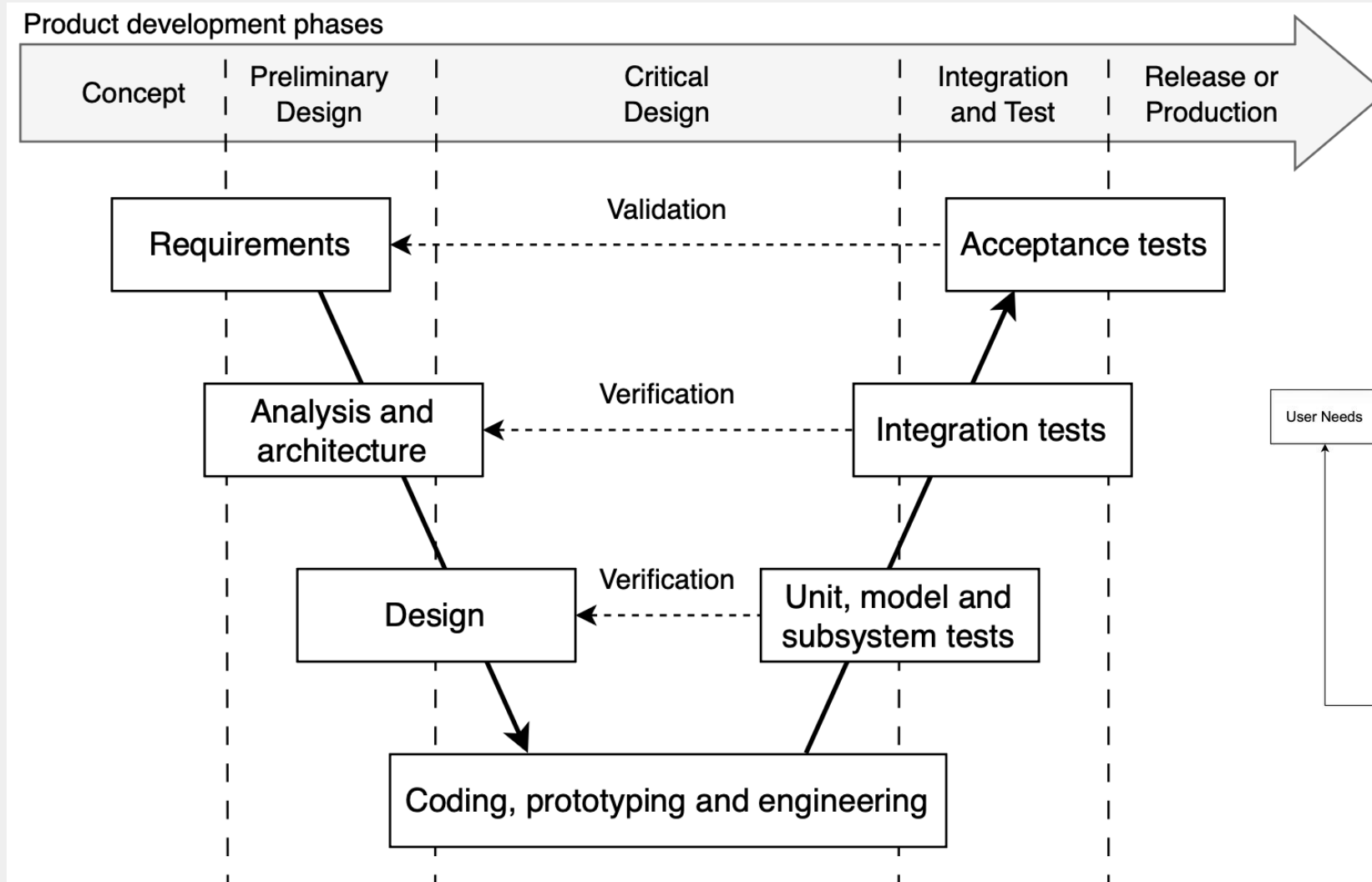
Vlad Știrbu

11.12.2025

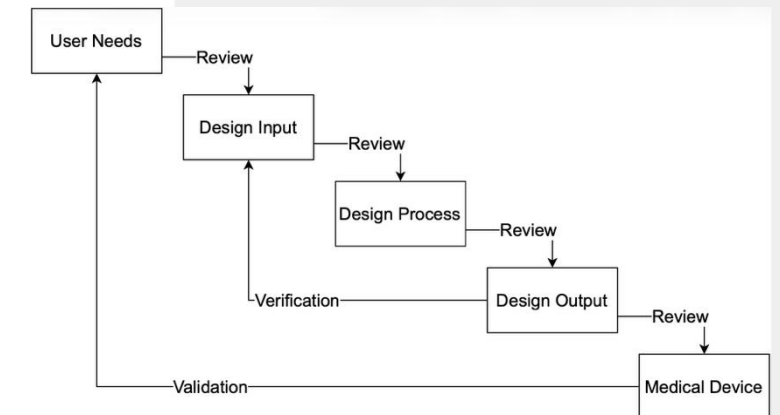




Traditional V-Model

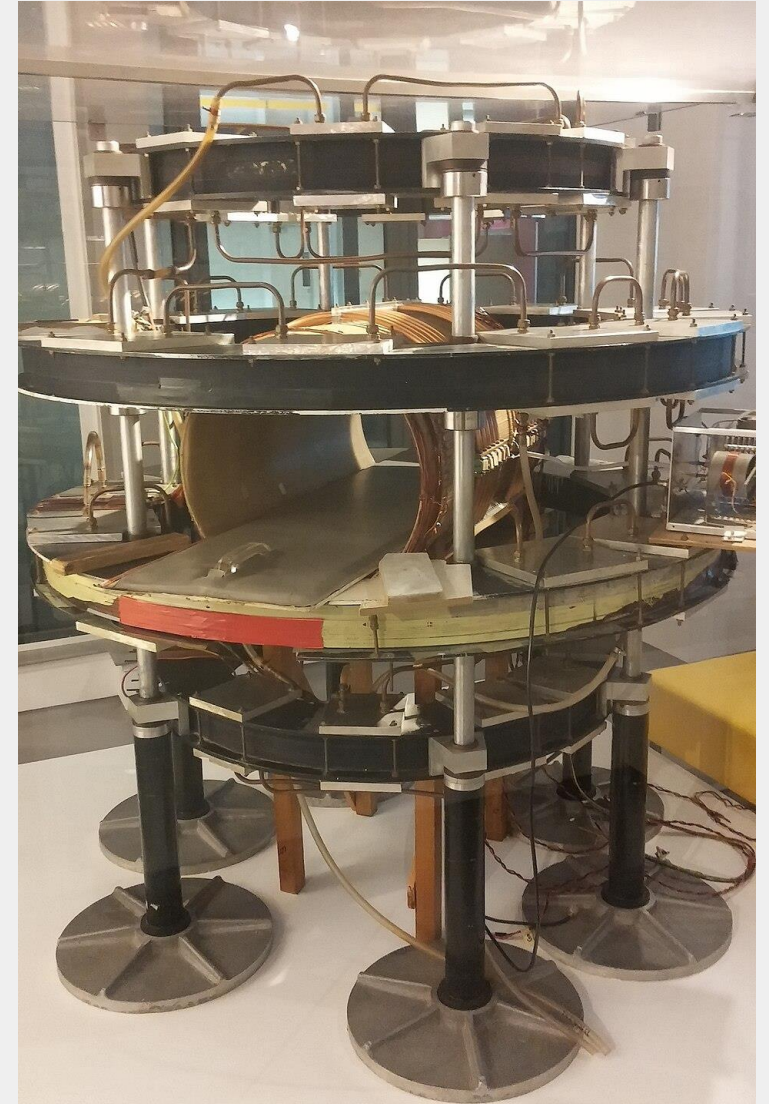
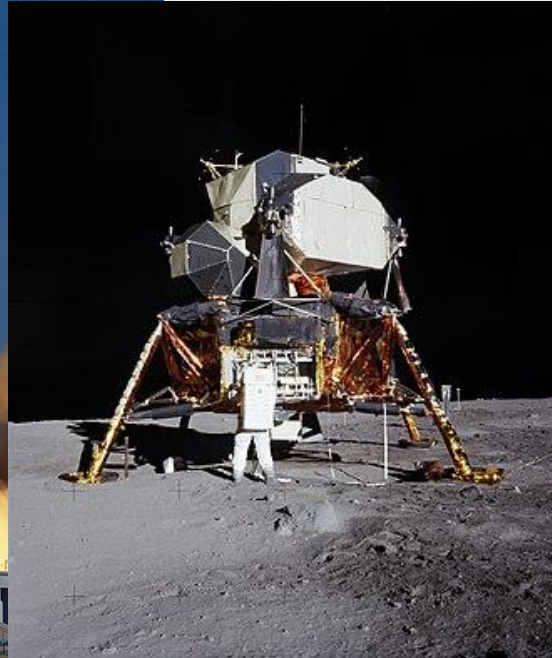
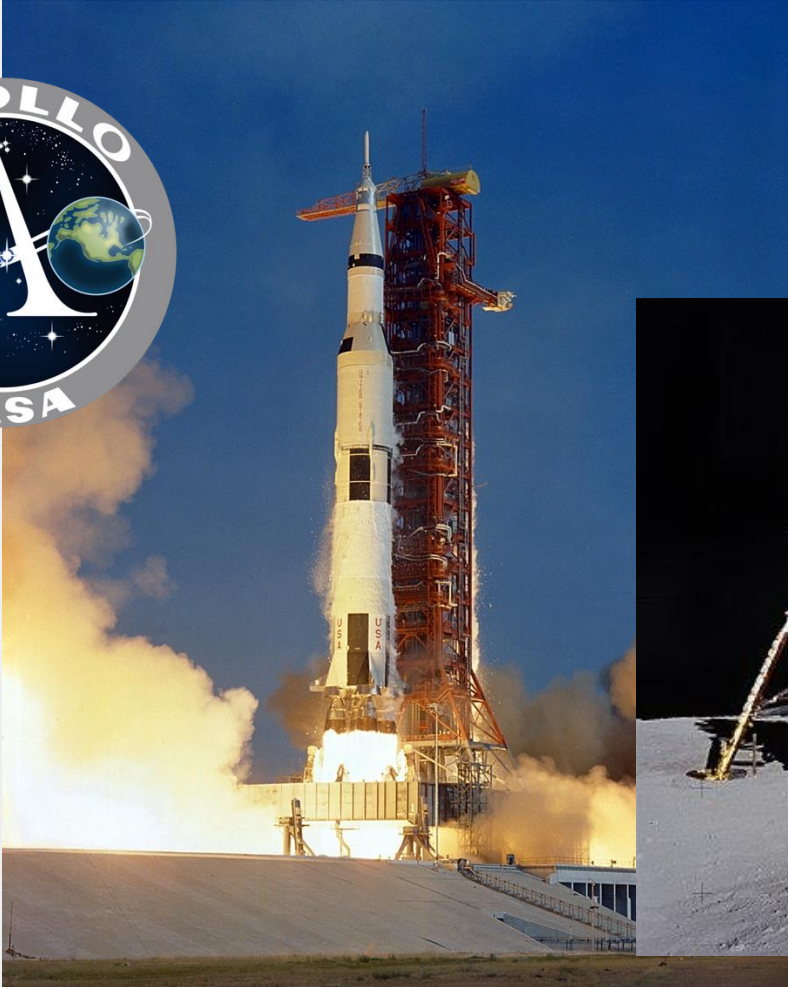


Design control





Success stories



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Software is eating the world

Marc Andreessen



<https://a16z.com/2016/08/20/why-software-is-eating-the-world/>



John Crickett • 2nd

CTO / VP Engineering / Head of Software Engineering

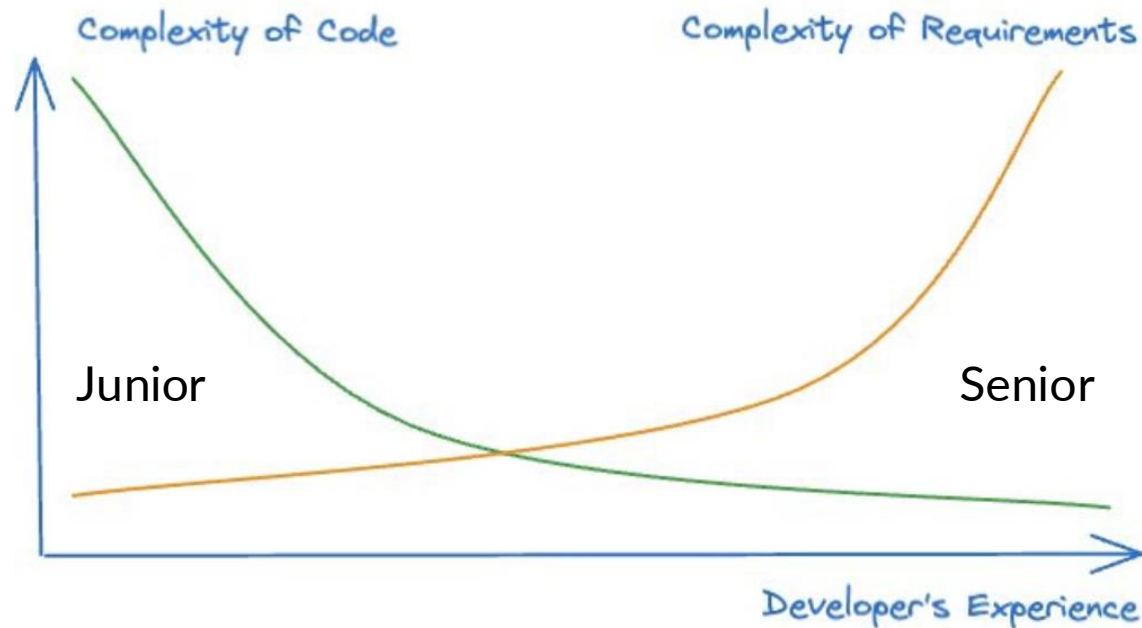
2yr •

[+ Follow](#) [...](#)

Junior developers take simple requirements and create complex code.

Senior developers take complex requirements and create simple code.

The difference is the senior developers' experience maintaining a complex system, teaches them the benefits of simple code.



https://www.linkedin.com/posts/johncrickett_junior-developers-take-simple-requirements-activity-7119940151601979392-0vRO/



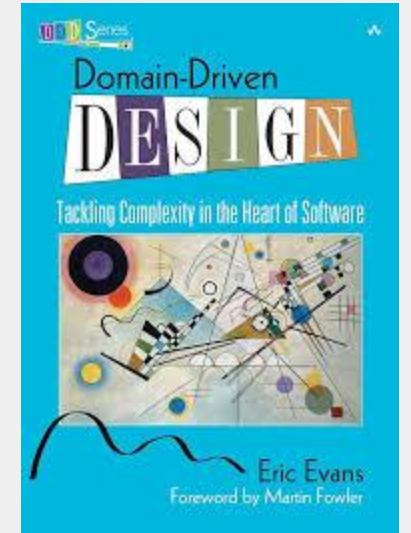
Toolbox – Domain driven design (DDD)

Core ideas

- Ubiquitous Language:
 - Shared vocabulary between developers and domain experts.
- Bounded Contexts:
 - Clear boundaries where a specific model applies (reduces complexity).
- Domain Model:
 - Business rules, concepts, and behaviors.

Consequences

- Aligns software with business needs
- Reduces accidental complexity
- Improves maintainability and clarity
- Supports modular and scalable architecture



Modeling the business **accurately** and structuring the software **around** that model



Toolbox - Architecture

C4 Model

- A visual model for describing software architecture
- Four levels:
 - Context → Containers → Components → Code
 - **UML the good parts**
- Structure, boundaries, responsibilities, and interactions
- Onboarding, communication, and system overviews
- Helps align teams on how the system is organized
- Diagrams evolve as architecture grows

Architecture decision records

- Capturing key technical decisions
- Describe options considered, trade-offs, and why a choice was made
- Historical context behind architecture
- Governance, audits, and long-term maintainability
- Understand reasoning, not just structure
- Evolve as decisions change or are superseded

C4 **describes** the architecture, ADRs explain **why** that architecture exists



Toolbox content - Executable requirements

Robot framework: automation engine + keyword DSL

```
Login Should Succeed
```

```
Input Text    username    alice
Input Text    password    secret
Click Button  Login
Page Should Contain  Welcome
```

- ✓ Automates tests
- ✓ Executes repeatable checks

Gherkin - human-friendly specification language for BDD

```
Scenario: Successful login
```

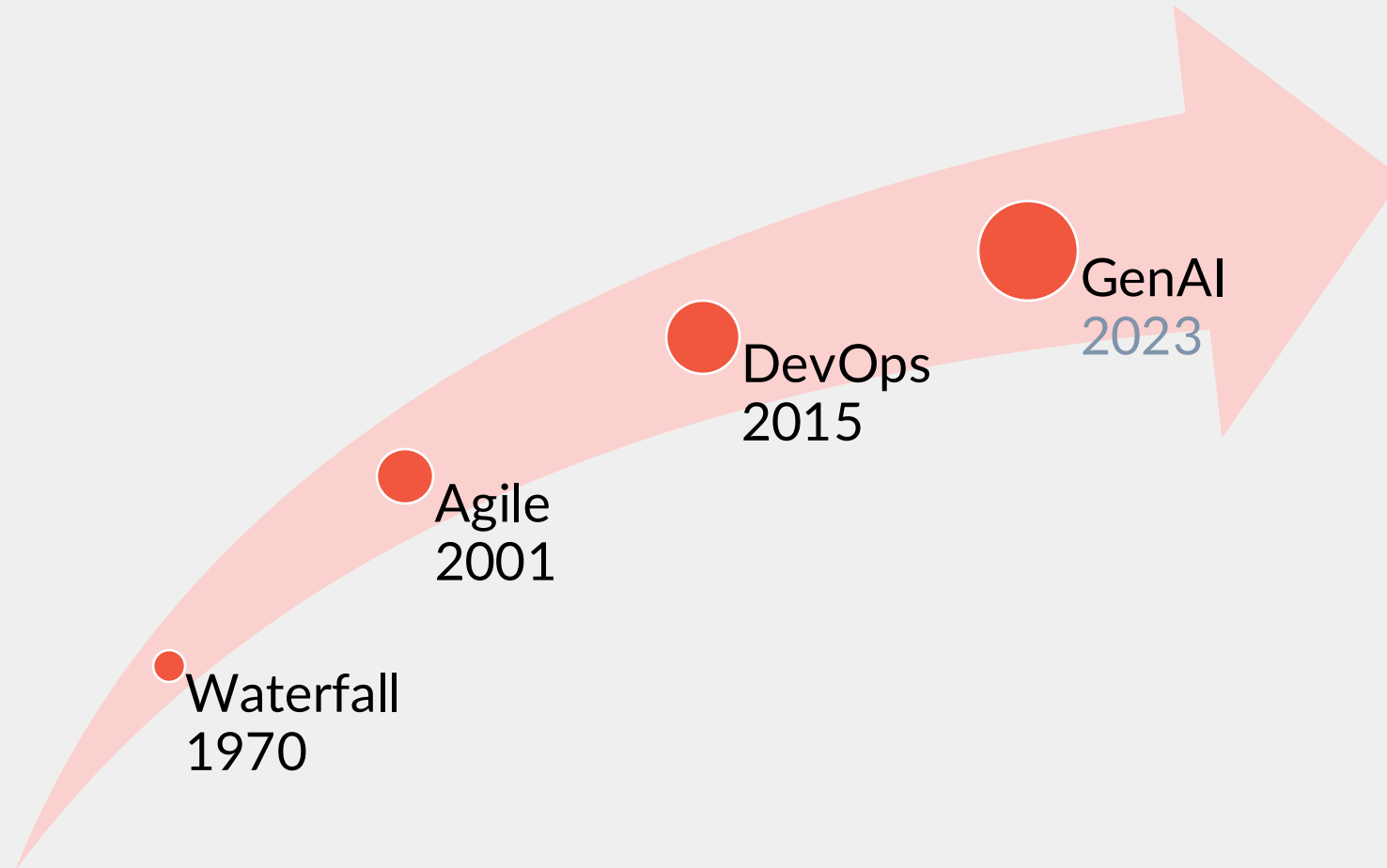
```
Given the user "alice" is on the login page
When she logs in with valid credentials
Then she sees the welcome message
```

- ✓ Describes expected behavior in plain English
- ✓ Helps business, QA, and developers agree on what the system should do

Gherkin **sets** expectations, Robot Framework **verifies** them

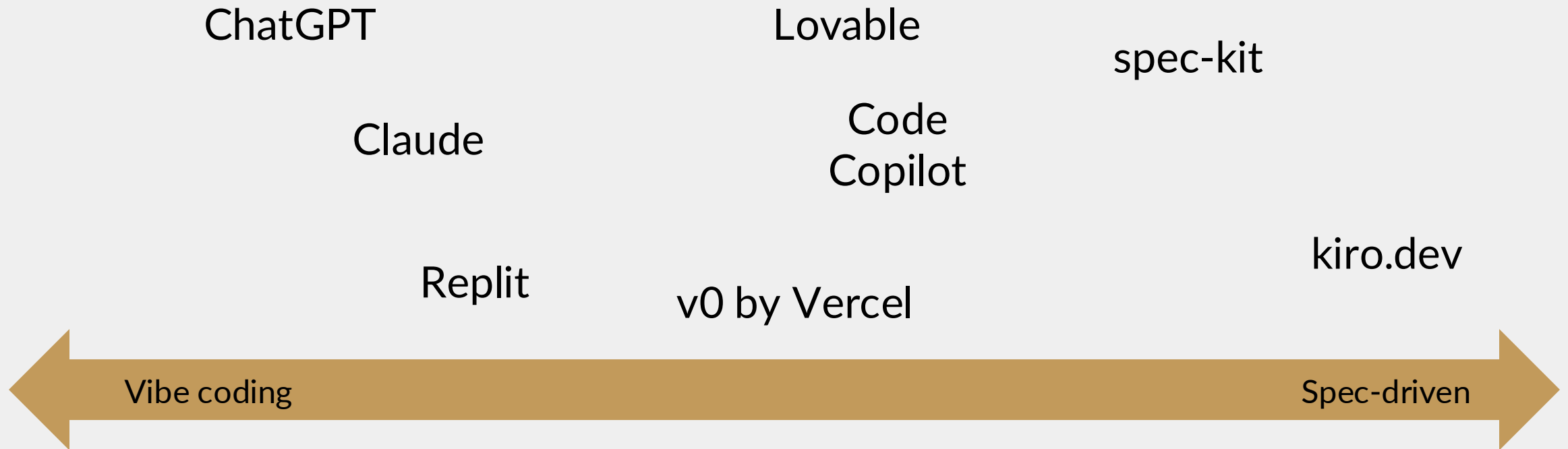


Software intensive products - methodologies



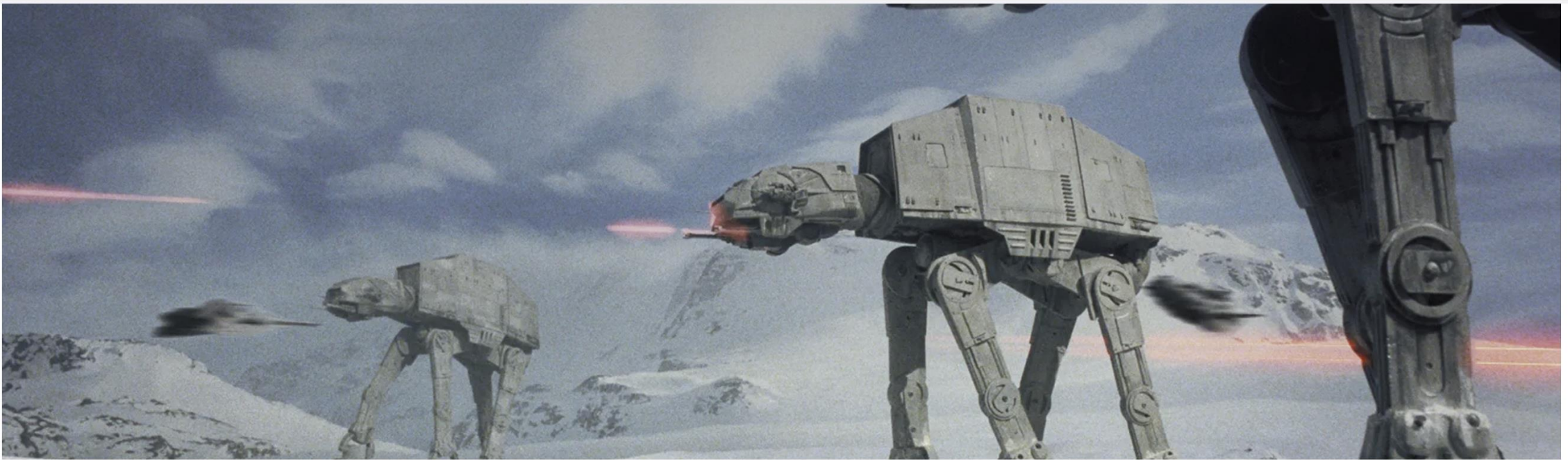


GenAI Landscape for Software Development



Informal programming style where you write code guided more by **intuition, flow, and rapid experimentation** than by strict upfront design or heavy formalism

Software-engineering approach where the primary artifact is a **machine-readable specification**—and the GenAI system **generates, maintains, and validates** the implementation around that spec.



Spec-Driven Development: The Waterfall Strikes Back



By **François Zaninotto**

November 12, 2025 • 7 min read

#agile #ai #architecture

Spec-Driven Development (SDD) revives the old idea of heavy documentation before coding — an echo of the Waterfall era. While it promises structure for AI-driven programming, it risks burying agility under layers of Markdown. This post explores why a more iterative, natural-language approach may better fit modern development.

The Rise of Specification

The Markdown Awakens

Revenge of the Project Manager

A New Hope

Conclusion

<https://marmelab.com/blog/2025/11/12/spec-driven-development-waterfall-strikes-back.html>




Julia Turc  @juliarturc · 16h




We need an official name for not-vibe-coding. My top candidates so far:

- boomer coding
- chewgy coding
- trad coding
- Coding with capital C

Suggestions are welcome.

 1.4K

 225

 2.6K

 180K



Gabor Varadi 

@Zhuinden

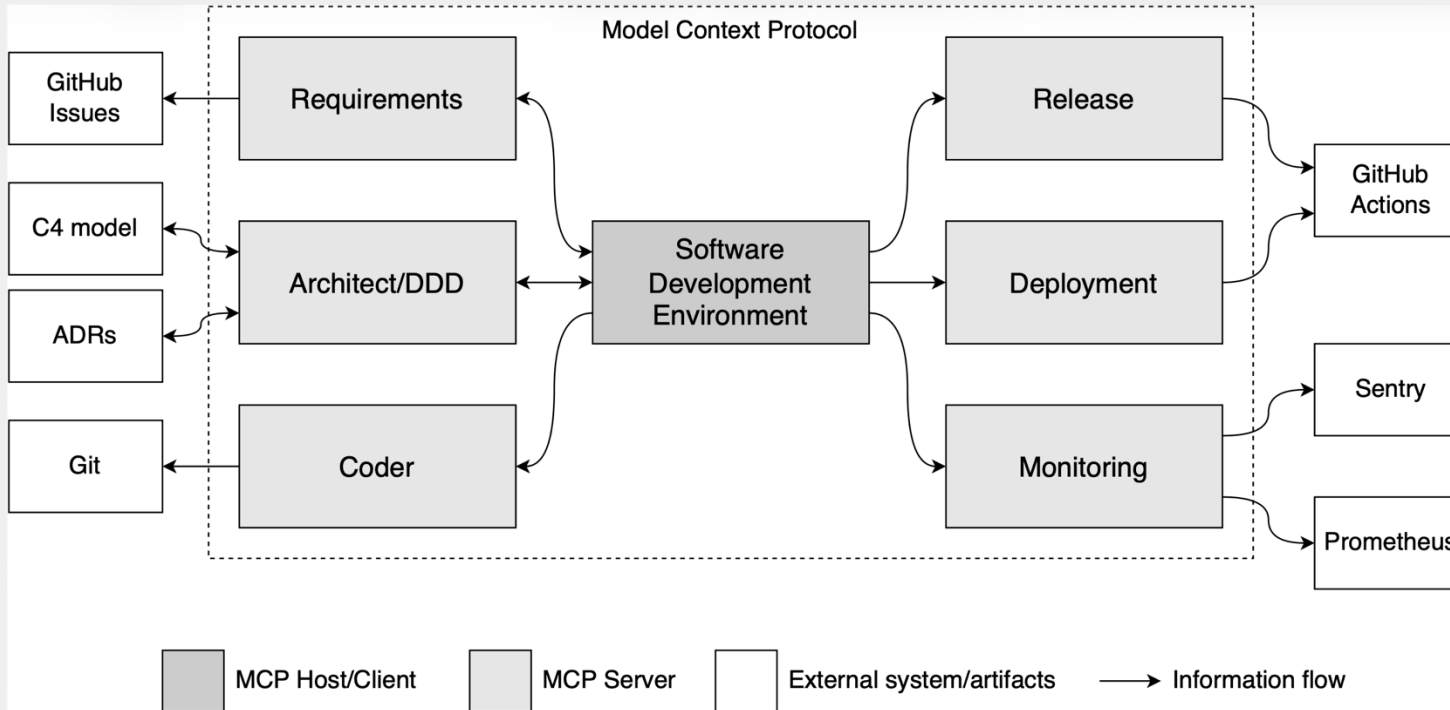
Replying to @juliarturc

"software engineering"

14:21 · 24 Oct 25 · **127** Views



ShiftUp implementation concept



<https://aaif.io/>



Linux Foundation Announces the Formation of the Agentic AI Foundation (AAIF), Anchored by New Project Contributions Including Model Context Protocol (MCP), goose and AGENTS.md

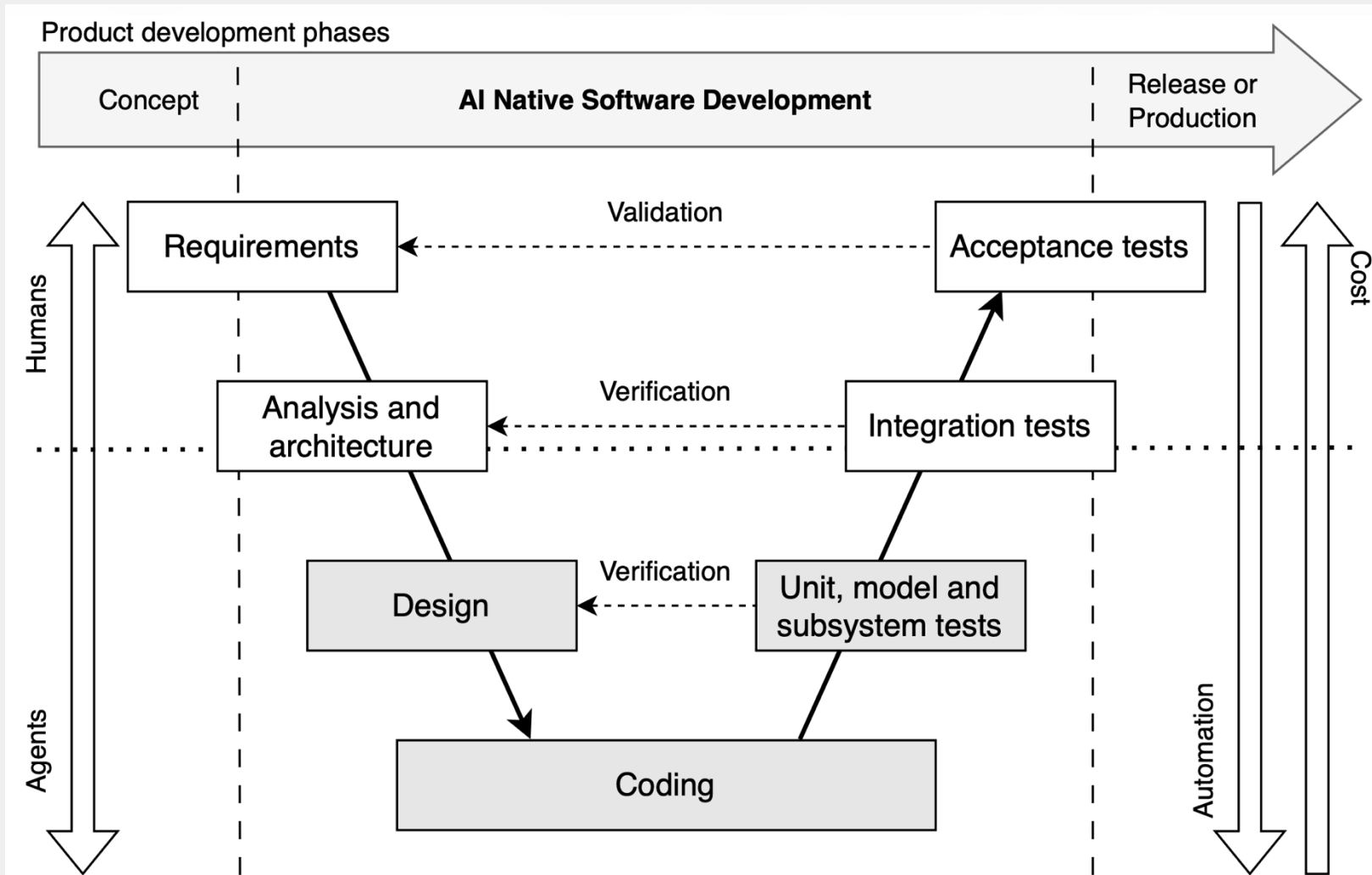
December 9, 2025

- Knowledge graphs = guardrails
- Requirements
 - Robot Framework, Gerkin
 - C4 model
 - Architecture Decision Records (ADR)

<https://arxiv.org/abs/2509.24485>



ShiftUp implications



- New abstractions
- Need to be explicit
- Explainability
- Governance



- ANSE - <https://www.jyu.fi/en/projects/anse>
- GENIUS - <https://www.jyu.fi/en/projects/genius>